ISRN:SICS-T--2000/06-SE
ISSN 1100-3154

ACOOR Rapport 1

TUFF: Systemöversikt och arkitektur

Martin Aronsson, Per Kreuger, Simon Lindblom och Per Holmberg

Email: piak@sics.se

2000-03-20

SICS Technical Report T2000:06

Nyckelord: planering, schemaläggning, villkorsprogrammering, omloppsplanering, cyklisk tid

Sammanfattning

TUFF är ett verktyg för strategisk planering av tågrörelser och resursomlopp (f.n. lok). Det unika med TUFF är inte att kunna utföra dessa planeringar var för sig, utan att de kan samverka. Detta är möjligt dels genom att de ingående komponenterna kan hantera vagare data, t.ex. tidsintervall i stället för fasta tider vilket är det vanliga annars bland kommersiella lösningar, dels att det finns en samordningsfunktion, som vi kallar samordningsagent, som samordnar olika planer från olika planeringsfunktioner. Denna samordnare är programmerbar via ett speciellt språk, så att ett stort problem kan splittras upp i mindre delar, och resultat från olika planeringar av dessa delar kan sedan kombineras och nya resultat genereras. Tanken med denna integrering av olika planeringsfunktioner är att åstadkomma globalt bättre lösningar, jämfört med när varje problem löses var för sig, genom att ta hänsyn till varje planeringsproblems krav tidigt i planeringscykeln.

ACOOR rapport 1

TUFF: Systemöversikt och arkitektur

Introduktion

TUFF är ett integrerat verktyg för planering av tågrörelser, både vad gäller spårlägen (när tåg kör på olika spåravsnitt) och lokomloppsplanering (vilka tåg som försörjs av olika lok), framtaget av SICS och SJ i samarbete. Arkitekturen är agentbaserad, bestående av ett grafiskt gränssnitt för formulering av planeringsproblemet, en koordinator-agent vilken formulerar delproblem och samordnar resultat, samt en eller flera agenter som är specialister på att utföra en viss sorts planering. För närvarande finns det två typer av specialister, omloppsplanerare och spårlägesplanerare. Ytterliggare en planerare, för personalplanering, är under utveckling.

Fokus för TUFF är integrationen av de olika planeringsproblemen. Tanken är att verktyget skall kunna användas i t.ex. anbudsskeden då olika affärsalternativ kan utvärderas vad gäller genomförandesätt. Verktyget kan också användas för att initialt göra strategiska avgöranden om ordningar mm mellan olika uppgifter, för att sedan traditionella verktyg inte skall suboptimera bort globalt bra lösningar.

För varje enskilt problem finns det idag olika kommersiella lösningar. Det kombinerade problemet betraktas som för komplext att lösa på denna nivå, och det blir nödvändigt att abstrahera och förenkla på ett sådant sätt att resultatet totalt sett är mer resurseffektivt än när en vattenfallsmodel med lokalt optimerande system används. Fokus för projektet är att dels formulera på vad sätt de olika problemen påverkar varandra, dels att formulera målet för de ingående planeringsagenterna och koordinatoragenten, samt att realisera dem.

En nyckelkomponent i dessa planeringsproblem är tid. Dels betraktas tid som cyklisk, dvs planering sker med t.ex. typveckor, vilket i sig komplicerar beräkningsmetoderna och beräkningsmodellerna jämfört med linjär tid, dels behövs olika finkornighet i olika sammanhang, vilket ställer krav på både den interna och externa representationen.

Problemformulering

Projektets huvudsyfte är att integrera olika planeringsprocesser hos SJ för att skapa globalt bättre lösningar än om lokalt optimerande system används i en vattenfallsmodell. Idag läggs huvudsakligen tidtabellen fast innan lokomlopp beräknas. Därefter tas personalplanerna fram. Tanken med TUFF är att i ett tidigare skede ta hänsyn till faktorer som påverkar fordonsomlopp och personalplanering, för att därigenom påverka förutsättningarna för tidtabelläggningen och därigenom nå ett bättre globalt resultat. Det är alltså inte frågan om att ersätta befintliga system, utan att komplettera dessa.

Ansatsen är att abstrahera och förenkla planeringsproblem för att minska komplexitet, samt att avgränsa delproblem som sedan planeras. Resultatet från dessa delplaner sammanfogas varefter nya planeringar genomförs, antingen på samma abstraktionsgrad eller genom att konkretisera resultatet från delplanerna. På detta sätt kan olika planeringsstrategier skapas, vilka skall kunna utryckas i ett speciellt språk, avpassat för detta ändamål. Följande är exempel på operationer som kan tänkas vara intressanta och som skall vara åtkomliga genom detta språk:

- Abstrahera planer, genom att tillåta större varians i t.ex. tider. Exempel: Starta någon gång mellan kl. 12 och 13
- Konkretisera planer, genom att tillföra ytterliggare begränsningar. Exempel: starta senast kl. 12:30
- Planera cykler, exempel: planera en typdag och gör likadant måndag till torsdag
- Dela upp problemet i affärer
- Dela upp problemet i geografiska zoner
- Dela upp problemet i olika typer, exempel: planera tidtabeller och lokomlopp var för sig och jämka samman resultaten
- Slå samman delplaner, av samma eller olika typer

Det finns många fler möjligheter. F.n. är dock detta under utveckling, och idag är enbart möjligheten att dela upp planeringen i spårlägesplanering och lokomlopp, samt att slå ihop planer möjliga. Arbete pågår för att kunna abstrahera resultat från planeringsagenterna, samt att formulera en första ansats till hur strategispråket skall se ut.

Abstraktion och konkretion av planer är ett av huvudområdena i TUFF. Stor vikt måste läggas vid hur abstraktion och konkretion görs, så att resultatet är konsistent, t.ex. om planer med olika tidsupplösning skall slås ihop, eller om en given plan skall konkretiseras genom att tidsintervall minskas. I TUFF betraktas planeringsoperationer som konkretioner av ursprungsplanen, vilka sedan kan abstraheras igen genom att återinföra intervall för vissa tidpunkter. De abstraktioner och konkretioner vi hitills har studerat och använt oss av baserar sig på olika typer av förändringar av tidsintervall snarare än olika tidsupplösning.

Fokus har i projektet hitills lagts på spårlägesplanering och lokomlopp, och f.n. klargörs hur sambanden mellan dessa två och personalplanering ser ut och hur dessa samband kan utnyttjas för att åstadkomma globalt bättre resultat.

Arkitektur

Systemet är idag implementerat i två olika programmeringssystem: Mozart och SICStus Prolog. Mozart är ett s.k. multiparadigm-språk, dvs det går att använda flera olika programmeringstekniker och -metodiker, bla objektorienterad programmering och funktionell programmering. SICStus Prolog är ett logikprogrammeringssystem med ett omfattande villkorsprogrammeringsbibliotek, vilket används för att lösa de olika beräkningarna som uppstår givet ett specifikt planeringsproblem.

Vidare består systemet av flera separata, parallella processer. Basen utgörs av en process implementerad i Mozart, där de olika planeringsagenterna exekveras i separata trådar, där en tråd i princip kan betraktas som en separat process internt i Mozartsystemet. För närvarande har varje planeringsagent en fristående underprocess, vilken är implementerad i SICStus Prolog. Mozartdelen av agenten är ansvarig för de beräkningar som bygger upp den matematiska formuleringen av planeringsproblemet, i termer av villkorsprogrammering, medan Prologprocessen löser detta matematiska problem. Kommunikationen mellan de olika

processerna sker över s.k. sockets, vilket är basen för internet-kommunikation, och med ett enkelt kommunikationsprotokoll.

Tanken är att varje agent skall exekvera i en egen tråd, men agentens eget behov avgör om den behöver stöd från något annat system. T.ex. så skulle en omloppsagent kunna utnyttja kommersiella lösare baserade på operationsanalytiska metoder, t.ex. CPLEX eller LPsolve.

En planering går till så att Mozartagenten skickar alla nödvändiga planeringsdata över socket-strömmen till Prologprocessen, vilken själv inte har någon egen uppfattning om tidsupplösning eller nät utan för varje planering får nya data. Planeringprocessen i Prologprocessen startas, och när denna terminerar skickas data tillbaka till Mozartagenten över socketströmmen. Svaret bearbetas något innan det skickas vidare till koordinatorn som lagrar svaren i interna databaser. Bearbetningen syftar till att ensa de olika formaten från de olika planeringsagenterna och det ursprungliga planeringsspecifikations-formatet. Detta för att det skall vara möjligt att utföra nya beräkningar, t.ex. andra planeringar, på resultatet från en planering. Det är således möjligt att spårlägesplanera resultatet av en spårlägesplan såväl som resultatet från en omloppsplanering.

De olika agenterna kommunicerar också med varandra via metodanrop. T.ex. frågar spårlägesagenten koordinatorn om nätdata genom att skicka ett metodanrop till koordinatoragenten som genom egen beräkning binder en variabel i anropet till det beräknade nätet.

Systemets olika delar

Systemet består som tidigare sagt av program och processer implementerade i två olika programmeringssystem. Bild 1 anger de olika faktiska delarna i systemet.

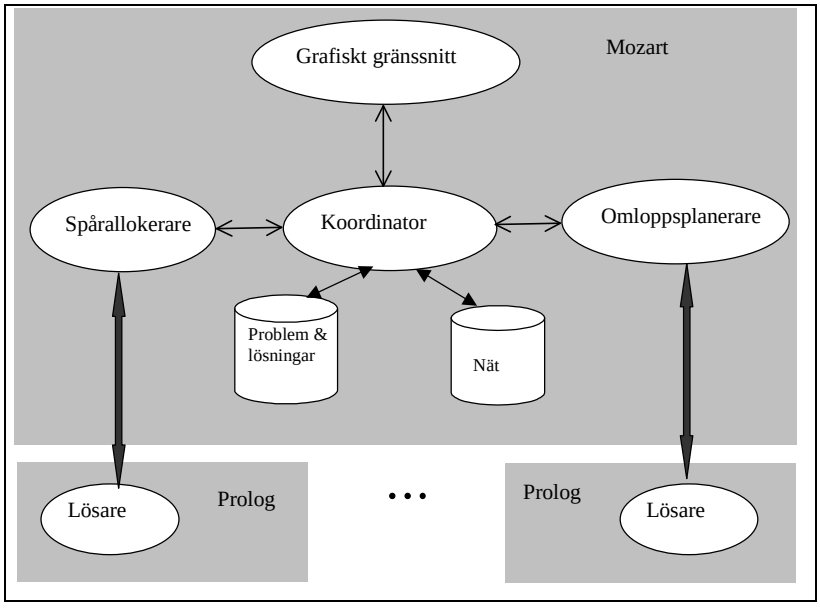

Bild 1

De tjockare linjerna representerar kommunikation över sockets, vilket i sin tur betyder att det är enkelt att lägga de olika processerna (de grå fälten) på fysiskt olika maskiner. De tunnare pilarna representerar olika trådar internt i Mozart-processen som kommunicerar med varandra, och de lite grövre representerar databas-hantering.

Vi inskränker oss i denna beskrivning till ett system bestående av en agent av varje typ.

Det grafiska gränssnittet

Det grafiska gränssnittet är ansvarigt för att ta emot kommandon och visa resultat för användaren. Det består av dels en större arbetsyta där olika problem skapas och skickas till planeraren, dels ett antal olika fönster som ger användaren möjlighet att se resultaten av planeringar i grafisk form, samt läsa in problemformuleringar och spara resultat på fil. Bild 2 visar huvudfönstret.

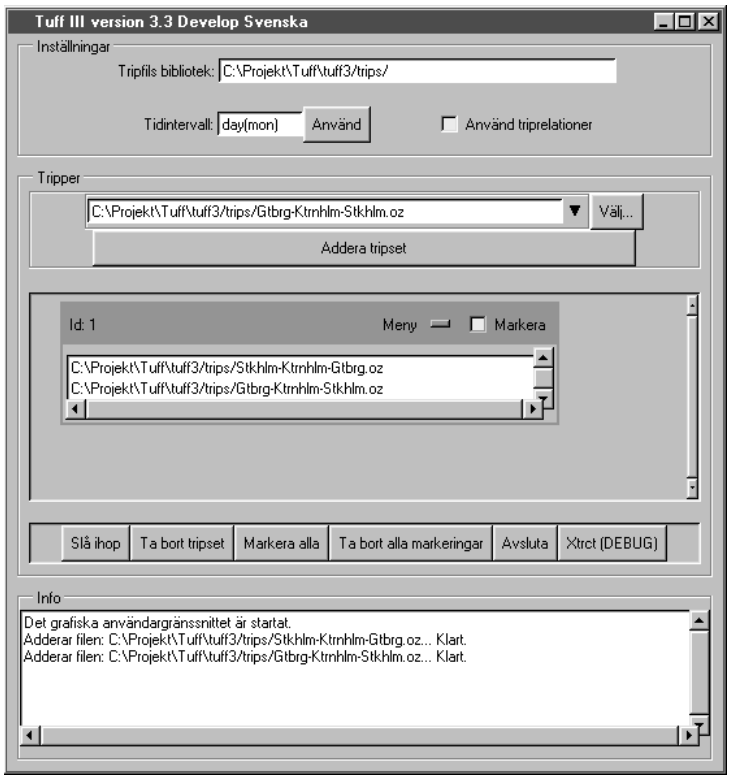

Bild 2

Varje agent har ett statusfönster kopplat till sig. Där loggas viss statusinformation och där det framgår om agenten är i vila eller om den jobbar. Bild 3 visar omloppsagentens fönster i vila.

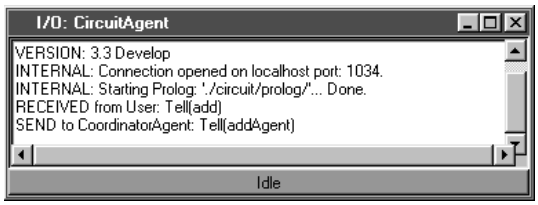

Bild 3

Ett resultat av att planera ett omlopp kan se ut som i Bild 4.

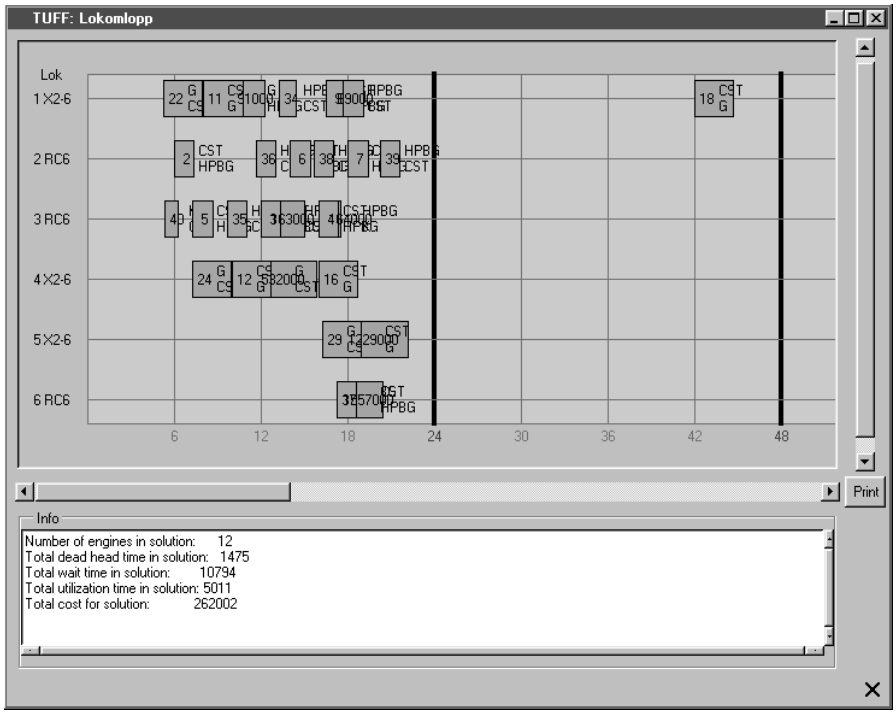

Bild 4

Omloppet anger ett visst loks rörelse i nätet, och varje liggande stapel anger en rörelse i nätet. Eftersom tiden är cyklisk finns det egentligen inget start eller slut på varje rad, som kallas för slinga, utan när ett lok kommer till slutet av raden så börjar det om igen. Varje rad anger alltså ett idealt förlopp för ett visst lok, i verkligheten tas förstås lok ur tjänst för service mm. Således representerar inte dessa slingor faktiska lokindivider. Cykeltidsgränsen representeras i grafiken med en grov vertikal linje, i detta fall ett typdygn.

En plan för spårlägen kan se ut som i Bild 5.

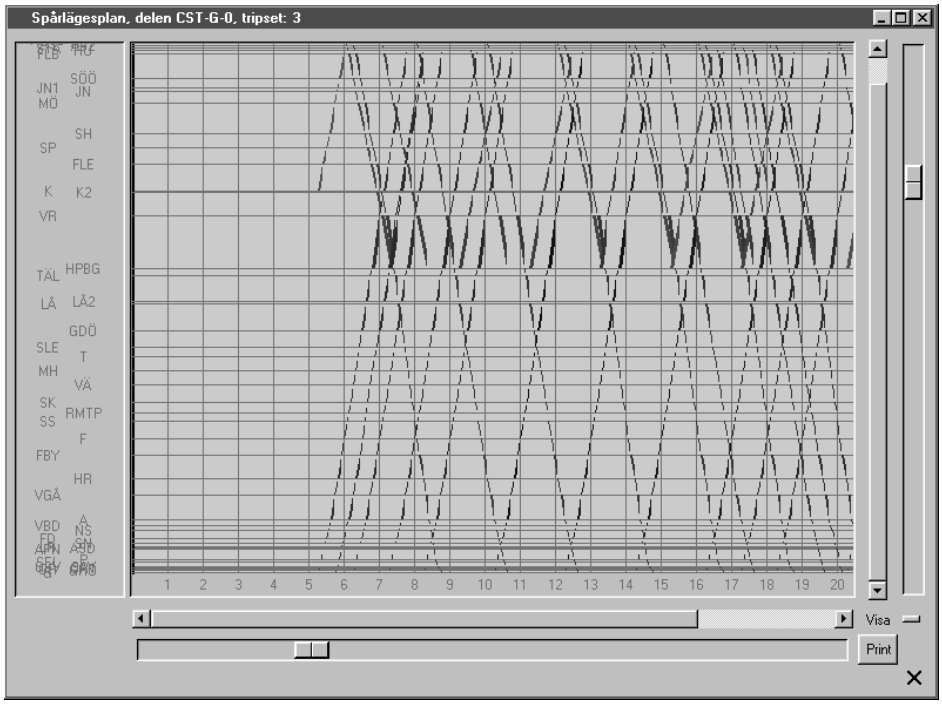

Bild 5

Längs vänstra kanten ligger orter i nätet, och horisontellt representeras tiden. De sneda strecken i grafen representerar tripper som traverserar respektive spårsegment mellan orterna.

Data och algoritmer

Basobjektet för varje planeringproblem är trippen. Trippen motsvarar en planerad tågrörelse, med en startplats och en slutplats, samt någon form av tidsbestämning (tidpunkt eller tidsfönster) för avgång, ankomst, traverseringstid, samt ev. andra villkor (omstigningar mm). Vidare är vägen som trippen skall ta genom spårnätet bestämd.

Tripperna genereras från en separat trippspecifikation. Specifikationerna är cykliska, vilket betyder att en viss tripp kan sägas återkomma med viss periodicitet, både på t.ex. timbasis som dygns- och veckobasis. Från dessa specifikationer genereras sedan en specifik mängd tripper beroende på vilken planeringsperioden som skall planeras. Om t.ex. specifikationen specificerar att tripp skall skapas två gånger dagligen måndag till fredag, och planeringscyckeln är satt till måndag tom tisdag, fås fyra tripper, två på måndag och två på tisdag. Formatet för dessa specifikationer är mycket generellt.

För spårlägesplaneringen krävs också en ytterliggare mängd data. Till varje tripp finns en mängd spårlägen, dvs de spårsegment som skall traverseras tillsammans med tidskrav för traverseringen av dessa spårsegment. Således har spårlägena liksom tripperna en specifikation av avgångstid, ankomsttid och traverseringstid, samt även en specifikation av det spårsegment som trippen traverserar.

Spårnätet består av spårsegment och stationer, där stationerna är noder och spårsegmenten är länkar.

Ett spårsegment är en spårlänk som förbinder två stationer. Varje spårsegment har ett antal egenskaper, varav de viktigaste är riktning (uppspår, nerspår eller dubbelriktad), samt per

tågtyp total traverseringstid och något som kallas för försprång eller headway. Traverseringstiden används för att schemalägga tripper som går åt olika håll på ett dubbelriktat spårsegment, headway används för att schemalägga tripper som går åt samma håll på ett dubbel- eller enkelriktat spårsegment. Vid dubbelspår antas det ena spåret vara allokerat till enbart upptrafik och det andra till enbart nertrafik dvs inga tripper kör åt motsatt håll på samma spårsegment.

En station är en resurs med kapacitet att rymma flera tripper samtidigt, dvs på en station kan tripper mötas och köra om varandra. Varje station har en begränsad kapacitet, som ej får överskridas vid något tillfälle.

Koordinatorn

Koordinatorn är ansvarig för samordning av planeringen, och är direkt kopplad till det grafiska gränssnittet. Användaren av verktyget "programmerar" koordinatorn, förser den med planeringsspecifikationer och interagerar med koordinatorn under problemlösningen. Planeringsproblemen i sig formuleras som trippspecifikationer i ett eget verktyg. Dessa trippspecifikationer är cykliska, typiskt en vecka, och specificerar hur enskilda tripper skall genereras, givet en viss tidsupplösning och ett visst intervall, t.ex. måndag till torsdag (se Data och algoritmer ovan). Upplösningen kan vara olika för olika delproblem.

Givet en mängd tripper kan koordinatorn skapa de data som behövs för de olika andra agenterna. För alla agenterna gäller att grunddata är de enskilda tripperna. För spårlägesagenten så genereras, förutom trippernas data, även data för spårlägena, vilket är tidskrav på allokering av enskilda spårsegment uttryckt i start- och sluttider samt riktning på spårsegmentet.

Koordinatorn fördelar olika planeringsuppgifter på olika agenter, och tar sedan emot resultatet från planeringen och lagrar denna i en databas. Därefter kan koordinatorn vidta olika åtgärder beroende på olika egenskaper hos resultatet. Tanken med koordinatorn är att den skall vara programmerbar i ett specifikt planeringsspråk. Detta språk skall innehålla operationer för att kunna kombinera olika problemformuleringar, skicka problem till olika specialistagenter vilka sedan returnerar resultatet från någon mer omfattande beräkning, varvid koordinatorn sedan analyserar resultatet. Beroende på vad som gjorts och vilket resultatet blev, kan koordinatorn t.ex. välja att behålla delar av resultatet och gå vidare med någon annan operation, eller kombinera resultatet med något annat resultat eller problem och återigen skicka detta till någon annan specialistagent. Genom att på detta sätt ge möjligheten att programmera upp globala lösningsprocedurer kan ett stort problem brytas ner till mindre delar, vilka kan planeras och sedan kombineras på olika sätt.

Detta språk är idag ej klart utan utreds för närvarande, och endast en del av operationerna finns tillgängliga, däribland möjligheten att slå ihop en mängd tripper med en annan mängd tripper till en ny mängd, vilken i sin tur kan planeras, samt vissa andra typer av abstraktioner.

Ett exempel på en enkel sekvens som kan genomföras är att från en problemspecifikation innehållande ett antal tripper med tidsintervall i avgångs- och ankomsttider generera ett omlopp. Extrahera från ett sådant omlopp de inbördes relationerna mellan avgångs- och ankomsttiderna som ges av vändningarna. Ge sedan spårlägesplaneraren i uppgift att skapa spårlägen baserat på ursprungsspecifikationen plus de ytterliggare relationerna mellan ankomst- och avgångstiderna som ges av omloppet. Tag sedan detta resultat, som är en tidtabell med fastlagda avgångstider, och skapa ett nytt omlopp. Detta sista steg sker idag med

samma omloppsplanerare som tidigare, men skulle lika gärna kunna vara en annan agent baserad på något optimerande verktyg.

Spårlägesplanering

spårlägesagenten är ansvarig för att allokera spårlägen på ett sådant sätt att tripperna inte krockar och att kapaciteten på stationerna inte överskrids. Detta sker för dubbelspår på ett sådant sätt att ett givet försprång, headway, upprätthålls för varje spårsegment. På enkelspår krävs att om tripperna traverserar spårsegmentet åt samma håll så upprätthålls försprånget, medan om de traverserar spårsegmentet åt olika håll så måste hela spårsegmentet traverseras av den ena trippen innan den andra kan allokera spårsegmentet.

På stationer kan tripper mötas och köra om. Varje given station har en given kapacitet i termer av antal tripper som samtidigt kan finnas på stationen, men i övrigt tas ingen hänsyn till stationens begränsningar (spårlängder mm). För varje tripp och varje station gäller också att stilleståndstiden på stationen måste hålla sig inom stipulerade krav i problemformuleringen, dvs avgångstid skall vara lika med ankomsttid och stilleståndstid adderat. Vid start och slut ställs krav på hur lång tid det tar innan tåget kan förflyttas bort från stationen, men ingen hänsyn tas till vart tåget egentligen tar vägen, utan bara att det inte upptar kapacitet på stationen, ”perrongplats” eller liknande platser som används för genomfart eller för ”affärsverksamhet”.

Alla tider beräknas cykliskt, dvs om planering sker för typdygn så identifieras klockslaget 24 med 0.

I det enkla exemplet nedan finns två tripper T_1, T_2 och T_3, som traverserar det enkelspåriga spårsegmentet x mellan stationerna A och B. T_1 och T_2 traverserar spårsegmentet i samma riktning och T_3 i motsatt riktning mot T_1 och T_2. Både A och B har en kapacitet av 2 tripper samtidigt. Följande start- och sluttidsintervall gäller för T_1, T_2 och T_3 i problemet:

	Start	Slut	Uppehåll A	Uppehåll B
T_1	5..15	10..20	-	1
T_2	5..8	9..12	-	1
T_3	6..12	12..18	1	-

Tabell 1 exempeldata för spårlägesplanering

Vidare är headway för spårsegmentet 1. Vi antar att schemat skall uttryckas i heltal.

Till problemet finns ett antal lösningar, bla följande.

	Start	Slut
T_1	6	11
T_2	5	9
T_3	11	17

Tabell 2 Lösning

I Bild 6 finns lösningen grafiskt. Den "skugga" som varje tripp lämnar efter sig representerar headway, och är inritad för samtliga tripper som en grå area. Ingen av dessa areor får överlappa någon annan area, vilket det underliggande villkorssystemet ser till. Detta är orsaken till att T_1 inte kan starta tidigare än 1 tidsenhet efter T_2. Vidare gäller för mötande tripper att areorna inte får korsa varandra, vilket systemet också håller reda på. I figuren kan T_3 starta tidigast vid tidpunkten 11.

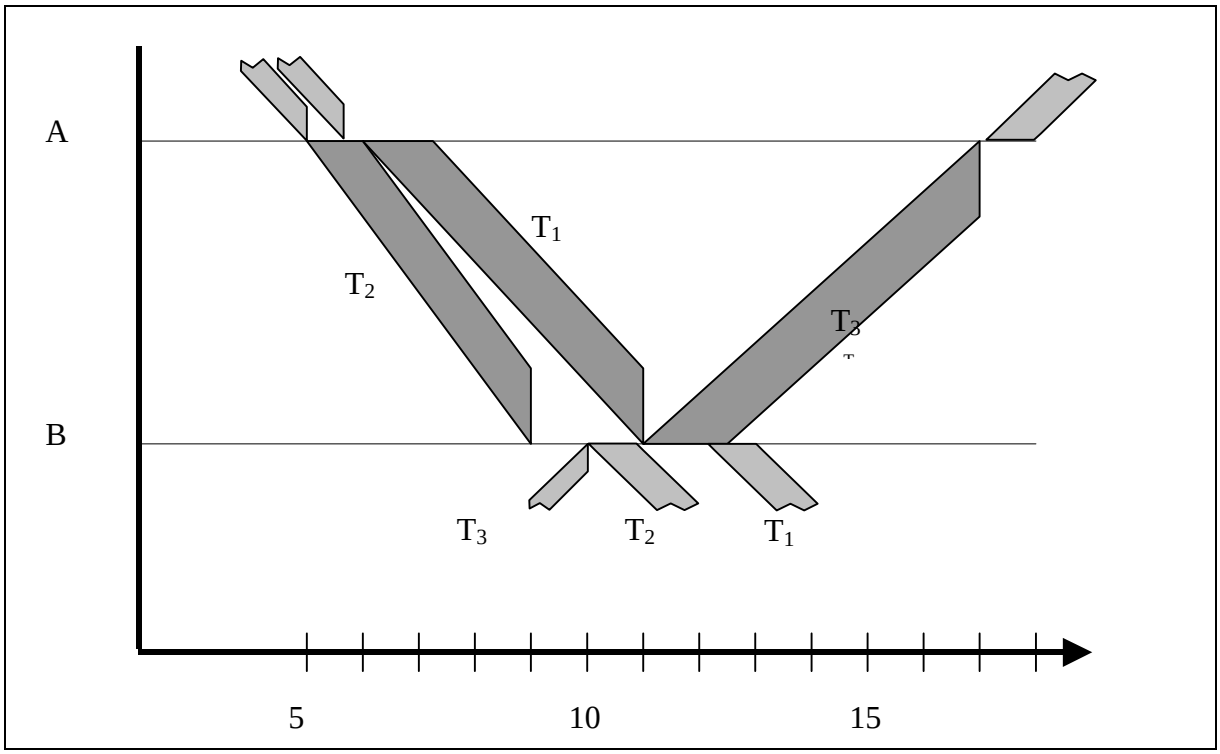

Bild 6

Lokomlopp

Lokomloppsagenten är ansvarig för att skapa omlopp som försörjer samtliga tripper med lok. Resultatet av planeringen är ett antal bindningar, vilka kallas vändningar, mellan tripperna. En vändning motsvarar att lok flyttas från en avslutad tripp till en annan tripp som därigenom försörjs med lokkapacitet. Planeringen sker även här med cirkulär tid (dvs typdygn, typveckor, etc), vilket betyder att dessa bindningar formar slingor. Till slingorna kan sedan lokindivider allokeras. Att allokera lokindivider ligger dock utanför den problemformulering vi arbetat med.

Målet för omloppsplaneraren är att identifiera de bindningar som leder till ett effektivt utnyttjande av lokresurser, givet att det finns spel i avgångs- och ankomsttiderna. Typiskt kan detta vara om en tripps avgångstid överlappar en annan tripps ankomsttid på en station, vilket med inskränkningar i tiderna leder till att de två tripperna kan bindas till varandra (loket vändas från den ankommande trippen till den avgående trippen).

Tekniskt sker planeringen genom att en $N \times N$ matris byggs upp, där N är antalet tripper. Varje element i matrisen motsvarar en möjlig vändning, med rader representerande

ankommande tåg och kolonner representerande avgående tåg. Algoritmen försöker sedan placera ut noll-skillda element i matrisen vilket motsvarar det antal lok som vänder (det kan vara fler än ett lok som vänder, t.ex. kan tunga godståg dras av två lok, och lok kan även passivt medfölja en tripp). När ett nollskilt element placeras någonstans i matrisen läggs ytterliggare villkor till som säkerställer att avgående tåg inte avgår tidigare än det ankommande tågets ankomst. När alla vändningar är fixerade har ett antal slingor bildats, dvs tripper ordnade sinsemellan så att de bildar ett cykliskt förlopp. Resultatet av omloppsberäkningen är dessa slingor, men ett förslag på fastställda värden för alla ankomst- och avgångstider beräknas också.

Detta liknar ett vanligt nätverksflöde (brukar bla finnas beskrivna i introduktionsböcker till operationsanalys som "network simplex", t.ex. "Introduktion to Operations Research", F. S. Hillier, G. J. Lieberman, McGrawHill), med den skillnaden att avgångs- och ankomsttider inte är fastställda initialt, vilket gör att traditionella metoder inom operationsanalysen inte kan användas. Vår ansats är baserad på villkorsprogrammering, med villkor som bygger upp det lösningsrum som lösningen skall hittas i. Lösningsalgoritmen försöker sedan utvärdera vilka av alla möjliga vändningar som skall väljas genom att utvärdera lokala kostnader för vändningsalternativen. Vi har även studerat en heltalsprogrammeringsformulering av problemet, men inte fått bättre prestanda med denna än med villkorsprogrammeringsvarianten.

Om en vändning sker från en tripp a som ankommer till en station A till en tripp b som avgår från en annan station B, så måste loket köras tomt från A till B. Denna transport kallas för ett självgående passiv transport eller ett "deadhead", och tar en viss tid att genomföra. För att kunna räkna på tidsavståndet mellan a:s ankomst och b:s avgång finns en avståndstabell som anger körtiden mellan A och B för varje typ av lok som förekommer. Egentligen är det fall där avgång och ankomst sker på samma ställe ett specialfall med transporttid satt till 0, eller den tid det tar att koppla loss loket, förflytta det på station och sedan koppla fast det igen framför ett nytt tåg.

Alla tider är cykliska, dvs om planering sker för typdygn så identifieras klockslag 24:00 med 00:00. Detta får bla som följd att relationer som ”före” och ”efter” inte är absoluta, t.ex. gäller för typdygn att klockan 23 ligger både före och efter klockan 1 (2 timmar åt ena hållet och 22 timmar åt andra hållet). Detta komplicerar en del beräkningar i systemet, och här har en hel del utveckling skett i TUFF-projektet.

Ett kraftigt förenklat exempel är följande. Antag att det finns tre platser A, B och C, och att periodens längd är 20 tidsenheter. Antag vidare att det finns 4 tripper, numrerade 1, 2, 3 och 4, med följande data där X..Y utläses som heltalsdomänen med alla heltal mellan X och Y:

Tripp	avgång	ankomst	avgångsstation	ankomststation
1	0..5	6..11	A	B
2	8..13	11..16	B	C
3	17..19	2..4	C	A
4	6..8	12..14	B	A

Tabell 3 exempeldata för omloppsgenerering

Notera att tiden är cyklisk, så att 18..1 i exemplet nedan skulle motsvara tidpunkterna 18, 19, 0 och 1. Matrisen för självgående passiva transporter (avståndstabellen) har följande utseende:

	A	B	C
A	1	6	5
B	6	1	3
C	5	3	1

Tabell 4 avståndsmatris mellan orterna A, B och C

Vi bortser vidare i detta enkla problem från att det kan finnas krav på olika loktyper samt att det krävs olika tid på olika stationer att vända loket (dvs koppla loss, rangera och koppla på loket igen), utan sätter vändnigstiden till på samtliga platser till 1, vilket syns i diagonalen i tabellen ovan.

En (av många) lösningar är då följande:

	1	2	3	4	Avgång	Ankomst
1		1			5	11
2			1		13	16
3	1				17	2
4				1	6	12

Tabell 5 Första omlopp

Egentligen finns det ett tidsintervall för t.ex. tripp 1 i avgångstid och ankomsttid på 3 tidsenheter, 3..5 respektive 9..11.

Tabell 5 Första omlopp ger följande Gantt-schema för de enskilda slingorna:

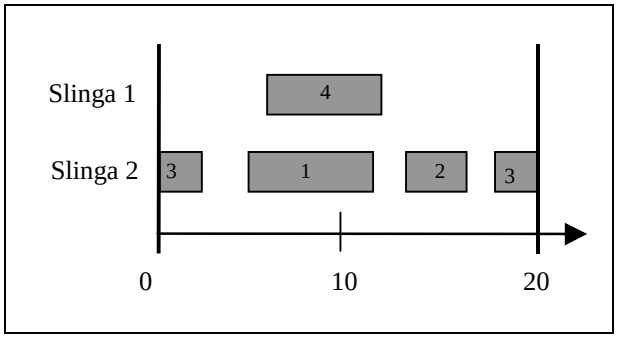

Bild 7

Notera att tripp 3 passerar cykeltidsgränsen, och således finns med både i början och i slutet på perioden. I ovanstående lösning behövs alltså två lok, ett per slinga, och att det i slinga 1

behövs en självgående passiv transport (deadhead) mellan plats A och B vilken inte är utritad i schemat.

En annan lösning med samma tider men med medföljande transport i stället för en självgående passiv transport mellan A och B är följande:

	1	2	3	4	Avg	Ank
1		1		1	5	11
2			1		13	16
3	1				17	2
4	1				6	12

Tabell 6 Andra omlopp

Notera att lok vänder in i tripp 1 både från tripp 3 och 4, samt att lok vänder från tripp 1 till både tripp 2 och 4. Det betyder att två lok medföljer tripp 1. Gantt-schemat för Tabell 6 Andra omlopp får följande utseende:

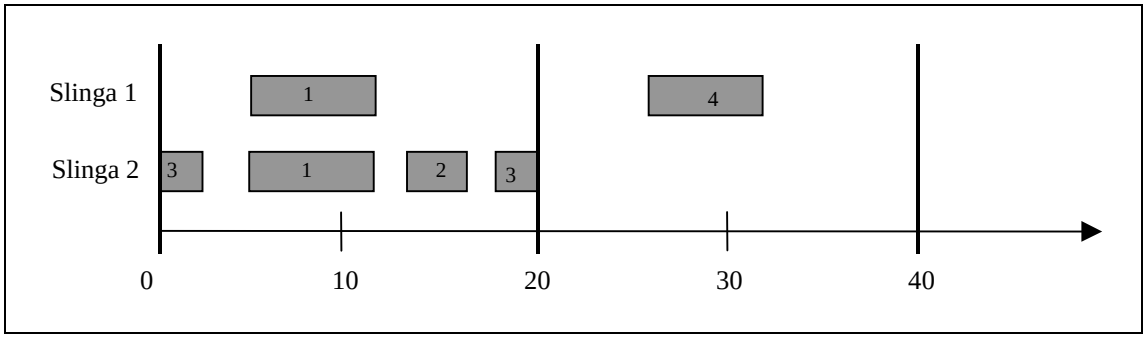

Bild 8

Här ligger två perioder efter varandra, och alltså behövs två lok för att realisera slinga 1 (de två perioderna skall ju realiseras i samtidigt). Slinga 2 behöver ett lok, vilket ger totalt tre lok, att jämföras med två lok i den första lösningen. Vilken som är bäst beror på vad som är dyrast, tre lok med en passiv transport mot två lok och en aktiv transport.

Notera också att Tabell 6 Andra omlopp innehåller möjligheter till olika slingor, t.ex. 1-2-3-1-4 som är ett alternativ till Gantt-schemat i Bild 8, med enbart en slinga som sträcker sig över tre perioder. Detta eftersom det finns ett val i framtagning av slingorna då det medföljer två lok med tripp 1. Slingan konstrueras i princip så ettorna i matrisen följs: starta t.ex. i rad 2 och följ raden till en kolumn som har en etta i sig. Läs av vilket kolumn-nummer (tripp) det är. Gå sedan in i den rad som har samma nummer som den nyss avlästa kolumnen hade och fortsätt till en kolumn som har en etta i sig, etc. Processen terminerar när man återkommer till samma skärning mellan rad och kolumn som man började på. Men om det finns två ettor på en rad så finns det en valmöjlighet vilken man väljer, vilket ger upphov till olika slingor.

Sammanfattning

Tuff är ett verktyg för strategisk planering av tågrörelser och resursomlopp (f.n. lok). Det unika med TUFF är inte att kunna utföra dessa planeringar var för sig, utan att de kan samverka. Detta är möjligt dels genom att de ingående komponenterna kan hantera vagare data, t.ex. tidsintervall i stället för fasta tider vilket är det vanliga annars bland kommersiella lösningar, dels att det finns en samordningsfunktion, som vi kallar samordningsagent, som samordnar olika planer från olika planeringsfunktioner. Denna samordnare är programmerbar via ett speciellt språk, så att ett stort problem kan splittras upp i mindre delar, och resultat från olika planeringar av dessa delar kan sedan kombineras och nya resultat genereras. Tanken med denna integrering av olika planeringsfunktioner är att åstadkomma globalt bättre lösningar, jämfört med när varje problem löses var för sig, genom att ta hänsyn till varje planeringsproblems krav tidigt i planeringscykeln.